Remarks

Claims 1-7, 10-12, 15, 17-23, 26-28, 31 and 47-48 are pending. All pending claims stand rejected under 35 USC §103.

Claim Rejections – 35 USC §103: The Examiner rejected Claim 1-7, 10-12, 15, 17-23, 26-28, 31 and 47-48 under §103(a) as being unpatentable over Brown (US Pub. No. 2002/0174180) in view of Ferrat (US Pub. No. 2005/0055382).

Claim 1 is directed to a coordinated push synchronization method and includes the following combination of elements.

a) detecting changes to a local application data store;
b) identifying a record affected by a detected change;
c) pushing the identified record to a remote application data store;
d) ascertaining whether the pushed record, in its current form as affected by the detected change, has already been replicated or deleted in the remote application data store in order to determine whether the remote application data store will be updated with the pushed record;
e) if not, updating the remote application data store with the pushed record and identifying the pushed record in the remote application data store as having been pushed from the local application data store to the remote application data store, otherwise ignoring the pushed record.

With respect to fourth element listed above, the Examiner asserts that Brown, paragraphs [0071] and [0080]-[0083], teaches "ascertaining whether the [pushed] record, in its current form as affected by the detected change, has already been replicated or deleted in the remote application data store in order to determine whether the remote application data store will be updated with the pushed record; if not, updating the remote application data store with the pushed record." The

Examiner Admits that Brown teaches that this act is performed before any data is pushed. The Examiner also indirectly admits that Brown fails to teach ignoring a **pushed** record if it is ascertained that the pushed record, in its current form as affected by the detected change, has already been replicated or deleted in the remote application data store.

To remedy Brown's deficiency, the Examiner mistakenly relies on Ferrat. The Examiner never asserts that Ferrat teaches ignoring a **pushed** record if it is ascertained that the pushed record, in its current form as affected by the detected change, has already been replicated or deleted in the remote application data store. The Examiner mistakenly asserts that Ferrat, paragraphs [0089]-[0091] teaches that a "record is pushed to the remote application data store prior to determining whether the record should be updated."

Claim 1 recites updating a record store and not updating a record. As such, the Examiner's assertion that Ferrat teaches pushing a record prior to updating the record is irrelevant. Assuming this to be a typographical error on the part of the Examiner, the cited paragraphs make no mention or suggestion of record is pushed to the remote application data store prior to determining whether the record **store** should be updated with the **pushed** record as recited in Claim 1. The cited paragraphs are reproduced as follows:

> [0088] 3.3 Push and Pull Mechanism
>
> [0089] One of the requirements that we would like to satisfy is the ability to do both "push" and "pull" in a single UniSync engine. For example, the ability to do a "push" for a spoke to move all the changes to the hub server and eventually resolve conflicts, and then do a "pull" to synchronize the hub server and the spoke database. These operations "push" and "pull" may be optional.
>
> [0090] The "push" and "pull" can be applied for all the commands that are described previously. UniSync engine provides the 3 basic commands snapshot, pointUpdate and continuousUpdate. They can "push" or "pull" depending of the requirement.
>
> [0091] These commands can be called directly from UniSync API. UniSync engine will support both "push" commands and "pull"

> commands at the same time. The user/application will decide to "push" or to "pull" and snapshot table for example depending on which site the command is executed. The outcome should be exactly the same for the UniSync engine.

Ferrat, paragraphs [0088]-[0091].

The cited passage discusses a synchronization environment with central hub server with several spoke clients. A spoke client can push changes to the hub server. The hub server resolves conflicts, if any. The spoke can then pull changes The conflicts resolved by the hub server relate to related changes received from other spoke clients. *See, e.g.,* Ferrat, paragraph [0082]. Ferrat makes no mention or suggestion that the hub server ascertains whether the pushed changes have already been replicated or deleted in a data store of the hub server prior to updating that data store. Further more, the Ferrat mentions nothing of ignoring the pushed changes if it is ascertained that the hub server's record store has already been updated with the pushed changes.

With respect to fifth element of Claim 1 listed above, the Examiner asserts that Brown teaches "identifying the [pushed] record in the remote application data store as a pushed record (paragraph 0066) and identifying the [pushed] record in the remote application data store as having been pushed from the local application data store to the remote application data store, otherwise ignoring the [pushed] record (paragraph 0071)."

Brown, paragraph [0066], is reproduced as follows:

> [0066] Metadata 330 shows the metadata for file 320 as stored within CSFS 335, part of client SA 337. (Although metadata are not shown for the other files and folders within folder 302, a person skilled in the art will recognize that such metadata exist.) In metadata 330, file 320 is shown as having a name (which is typically not encrypted, although the name can be encrypted in an alternative embodiment of the invention), a CID of 0x62, a SID of 0x2A, a FSI of 35, the change time of the file, and the flags used in the synchronization process (**such as identifying**

> **metadata items that need to be pushed to the server**). Note that metadata 330 is not shown to store the data of file 320, which is stored in the native operating system of computer 130 within the folder structure, as expected.

Brown, paragraph [0066] (emphasis added). Brown's paragraph 66 mentions nothing of identifying a record in the remote application data store as a pushed record. It simply discusses a flag that identifies metadata (not a record) that needs to be pushed to a server.

Brown, paragraph [0071], is reproduced as follows:

> [0071] The client polls the server for changes by other clients by passing its current CSI to the server in a sync polling call. If the CSI matches the server account's SSI value, then the client is up to date with the server. Otherwise the client SA requests server synchronization data (SSD). The SSD contains the following data:

Brown, paragraph [0071]. Brown's paragraph 71 mentions nothing of identifying a record in the remote application data store as having been pushed from the local application data store to the remote application data store

Consequently, the combination of Brown and Ferrat fail to teach or suggest a method that includes:

a) ascertaining whether the pushed record, in its current form as affected by the detected change, has already been replicated or deleted in the remote application data store in order to determine whether the remote application data store will be updated with the pushed record; and
b) if not, updating the remote application data store with the pushed record and identifying the pushed record in the remote application data store as having been pushed from the local application data store to the remote application data store, otherwise ignoring the pushed record.

For at least these reasons, Claim 1 is patentable over Brown and Pivowar. Claims 2-4 and 47 are thus also felt to distinguish over those references based on their dependency from Claim 1.

Claim 5 is directed to a coordinated user-initiated synchronization method and includes the following combination of elements:

1. detecting changes to a local application data store;
2. identifying a record affected by a detected change;
3. ascertaining whether the identified record, in its current form as affected by the detected change, was pushed to the local application data store from a remote application data store; and
4. if not, synchronizing the remote application data store with the local application data store.

With respect to the third element listed above, the Examiner admits that Brown fails to teach or suggest "ascertaining whether the identified record, in its current form as affected by the detected change, was pushed to the local application data store; and if not, synchronizing the remote application data store with the local application data store." ***ONCE AGAIN*** it is noted that the Examiner is mischaracterizing that which Claim 5 recites – specifically – "ascertaining whether the identified record, in its current form as affected by the detected change, was pushed to the local application data store **<u>from a remote application data store</u>**" (emphasis added). The Examiner's discussion of Claim 5 ignores this emphasized portion of Claim 5. As such, the rejection is improper.

Nonetheless, the Examiner mistakenly relies on Ferrat to remedy Brown's deficiency. The Examiner once again, asserts that Ferrat, paragraphs [0089]-[0091] teach ascertaining whether the identified record, in its current form as affected by the detected change, was pushed to the local application data store; and if not, synchronizing the remote application data store with the local application data store.

The cited paragraphs are reproduced as follows:

> [0088] 3.3 Push and Pull Mechanism
>
> [0089] One of the requirements that we would like to satisfy is the ability to do both "push" and "pull" in a single UniSync engine. For example, the ability to do a "push" for a spoke to move all the changes to the hub server and eventually resolve conflicts, and then do a "pull" to synchronize the hub server and the spoke database. These operations "push" and "pull" may be optional.
>
> [0090] The "push" and "pull" can be applied for all the commands that are described previously. UniSync engine provides the 3 basic commands snapshot, pointUpdate and continuousUpdate. They can "push" or "pull" depending of the requirement.
>
> [0091] These commands can be called directly from UniSync API. UniSync engine will support both "push" commands and "pull" commands at the same time. The user/application will decide to "push" or to "pull" and snapshot table for example depending on which site the command is executed. The outcome should be exactly the same for the UniSync engine.

Ferrat, paragraphs [0088]-[0091].

The cited passage discusses a synchronization environment with central hub server with several spoke clients. A spoke client can push changes to the hub server. The hub server resolves conflicts, if any. The spoke can then pull changes The conflicts resolved by the hub server relate to related changes received from other spoke clients. *See, e.g.,* Ferrat, paragraph [0082]. Ferrat makes no mention or suggestion that the hub server ascertains whether an record identified as having been changed, in its current form as affected by the detected change, was pushed to the hub server from a spoke client and, if not, local application data store from a remote application data store, and, if not, synchronizing the spoke client with the hub server.

Consequently, the combination of Brown and Ferrat fail to teach or suggest a method that includes:

a) ascertaining whether the identified record, in its current form as affected by the detected change, was pushed to the local application data store from a remote application data store; and
b) if not, synchronizing the remote application data store with the local application data store.

For at least these reasons, Claim 5 is felt to distinguish over Brown. Claims 6, 7, and 48 are thus also felt to distinguish over Brown based on their dependency from Claim 5.

Claim 10 is directed to a coordinated push and user-initiated synchronization method and includes the following combination of elements.

1. detecting changes to a local application data store;
2. identifying a first record in the local application data store affected by a detected change;
3. pushing the first record to a remote application data store;
4. ascertaining whether the pushed record, in its current form as affected by the detected change, has already been replicated in or deleted from the remote application data store and, if not, updating the remote application data store with the pushed record;
5. detecting changes to the remote application data store;
6. identifying a second record in the remote application data store affected by a detected change;
7. ascertaining whether the second record, in its current form as affected by the detected change, has already been pushed into the remote application data store in order to determine whether the remote application data store will be updated with the pushed record and, if not, synchronizing the remote application data store with the local application data store, otherwise ignoring the pushed record.

As with Claim 1, Brown and Ferrat fail to teach a method that includes "ascertaining whether the pushed record, in its current form as affected by the detected change, has already been replicated in or deleted from the remote application data store and, if not, updating the remote application data store with the pushed record"

For at least these reasons, Claim 10 is felt to distinguish over Brown and Ferrat. Claims 11, 12, and 15 are thus also felt to distinguish over Brown and Ferrat based on their dependency from Claim 10.

Claim 17 is directed to a computer readable medium having instructions for performing the method steps of Claim 1. For the same reasons Claim 1 distinguishes over Brown and Ferrat so does Claim 17. Claims 18-20 are thus also felt to distinguish over Brown and Ferrat based on their dependency from Claim 17.

Claim 21 is directed to a computer readable medium having instructions for performing the method steps of Claim 5. For the same reasons Claim 5 distinguishes over Brown and Ferrat so does Claim 21. Claims 22 and 23 are thus also felt to distinguish over Brown and Ferrat based on their dependency from Claim 21.

Claim 26 is directed to a computer readable medium having instructions for performing the method steps of Claim 10. For the same reasons Claim 10 distinguishes over Brown and Ferrat so does Claim 26. Claims 27, 28, and 31 are thus also felt to distinguish over Brown and Ferrat based on their dependency from Claim 26.

Conclusion: All pending claims are felt to be in condition for allowance. The foregoing is believed to be a complete response to the outstanding office action.

Respectfully submitted,
Richard Detweiler, et al

By /Jack H. McKinney/

Jack H. McKinney
Reg. No. 45,685

November 15, 2006